Definiciones recursivas

Lógica

Recursión

Dado un conjunto inductivo, sabemos exactamente cómo se construyen sus elementos.
Esta información sirve para:

- Probar propiedades de sus elementos (inducción)
- Definir funciones sobre sus elementos (recursión)

¿Qué es una función?

Una función es una relación que

asocia un único elemento del codominio a cada elemento del dominio.

Una función es un mecanismo de cómputo que

para cada entrada (valor del dominio) devuelve *efectivamente* un mismo valor del codominio.

***Efectivamente* significa**

- Para cualquier elemento del dominio hay una imagen.
- El cómputo termina para cualquier elemento del dominio.

Esquema de Recursión Primitiva para $\mathbb{N}$

$\mathbb{N} \subseteq \mathbb{R}$ definido inductivamente por:

i. $0 \in \mathbb{N}$
ii. Si $n \in \mathbb{N}$, entonces $S(n) \in \mathbb{N}$.

ERP para $\mathbb{N}$ (informal)

Sea B un conjunto cualquiera. Entonces, para definir una única función $F : \mathbb{N} \to B$ basta un conjunto de ecuaciones como el siguiente:

i. $F(0) = \ldots$
ii. $F(S(n)) = \ldots F(n) \ldots n \ldots$

Esquema de Recursión Primitiva para $\mathbb{N}$

- Método que se aplica para definir funciones sobre objetos de $\mathbb{N}$
- Usa el conocimiento de cómo se generan los objetos de $\mathbb{N}$

Al dar una definición inductiva mostramos cómo identificar (o “construir”) cada objeto del conjunto mediante las reglas dadas.

Aplicaciones del ERP para $\mathbb{N}$

Definir

- Factorial de k
- Suma de los primeros k naturales ($\sum_{1 \leq i \leq k} i$)
- Sumarle k a ...
- Multiplicar k por ...
- Elevar k a la ...

Esquema de Recursión Primitiva para un Conjunto Inductivo

Sea A un conjunto definido inductivamente. Para definir una función $f : A \to B$ alcanza con proporcionar ecuaciones que determinen

- el valor de f para los objetos de A obtenidos de aplicar cláusulas base
- el valor de f para los objetos de A obtenidos de aplicar cláusulas inductivas, utilizando el valor de f en el (los) objeto(s) anterior(es) y también el (los) objeto(s) anterior(es) (*llamadas recursivas*)

Esquema de Recursión Primitiva para L_1

$L_1 \subseteq \{a, b\}^*$ definido inductivamente por:

i $a \in L_1$

ii Si $w \in L_1$, entonces $bwb \in L_1$.

ERP para L_1 (informal)

Sea B un conjunto cualquiera. Entonces, para definir una única función $F : L_1 \rightarrow B$ basta un conjunto de ecuaciones como el siguiente:

i $F(a) = \ldots$

ii $F(bwb) = \ldots F(w) \ldots w \ldots$

Esquema de Recursión Primitiva para Σ^*

$\Sigma^* \subseteq \Sigma^*$ definido inductivamente por:

i $\varepsilon \in \Sigma^*$

ii Si $w \in \Sigma^*$ y $x \in \Sigma$, entonces $xw \in \Sigma^*$.

ERP para Σ^* (informal)

Sea B un conjunto cualquiera. Entonces, para definir una única función $F : \Sigma^* \to B$ basta un conjunto de ecuaciones como el siguiente:

i $F(\varepsilon) = \ldots$

ii $F(xw) = \ldots F(w) \ldots w \ldots x \ldots$

Aplicaciones del ERP para Σ^*

Definir

- Largo $: \Sigma^* \to \mathbb{N}$
- EsVacía $: \Sigma^* \to \{0, 1\}$
- Espejo $: \Sigma^* \to \Sigma^*$

Para qué necesitamos el ERP?

$f_i : \mathbb{N} \to \mathbb{N}$ sin ERP

$$
\begin{aligned}
f_1(0) &= 1 \\
f_1(n+1) &= \\
f(n+2) &- 2
\end{aligned}
$$

$$
\begin{aligned}
f_2(0) &= 1 \\
f_2(n+2) &= f_2(n) + 1 \\
f_2(n+2) &= f_2(n) - 3
\end{aligned}
$$

- Para algunos argumentos nunca termina el cálculo.
- Para algunos argumentos devuelve dos valores diferentes.

Las Funciones

- Son totales (Cubren todo el dominio - Exhaustividad).
- Devuelve un único valor para un argumento dado (No superposición).
- Para cualquier argumento devuelven un valor

Formalización del ERP para $\mathbb{N}$

Hipótesis

Sea B un conjunto, y

- un elemento $f_0 \in B$, y
- una función $f_s : \mathbb{N} \times B \to B$

Tesis

Entonces existe una única función $F : \mathbb{N} \to B$ tal que

i. $F(0) = f_0$
ii. $F(S(n)) = f_s(n, F(n))$

Formalización del ERP para $\mathbb{N}$: ejemplo

Una opción

$$\begin{aligned} \textsc{fact}(0) &= 1 \\ \textsc{fact}(S(n)) &= S(n) \times \textsc{fact}(n) \end{aligned}$$

Otra opción

$$\begin{aligned} f_0 &= 1 \\ f_s(n, r) &= S(n) \times r \end{aligned}$$

Formalización del ERP para L_1

Hipótesis

Sea B un conjunto, y

- un elemento $f_a \in B$, y
- una función $f_s : L_1 \times B \to B$

Tesis

Entonces existe una única función $F : L_1 \to B$ tal que

i $F(a) = f_a$

ii $F(bwb) = f_s(w, F(w))$

Formalización del ERP para Σ^*

Hipótesis

Sea B un conjunto, y

- un elemento $f_\varepsilon \in B$, y
- una función $f_s : \Sigma \times \Sigma^* \times B \to B$

Tesis

Entonces existe una única función $F : \Sigma^* \to B$ tal que

i $F(\varepsilon) = f_\varepsilon$

ii $F(xw) = f_s(x, w, F(w))$

Formalización del ERP para Σ^*: ejemplo

Una opción

$$\begin{aligned}
\mathrm{ESPEJO}(\varepsilon) &= \varepsilon \\
\mathrm{ESPEJO}(xw) &= x\ \mathrm{ESPEJO}(w)\ x
\end{aligned}$$

Otra opción

$$\begin{aligned}
f_\varepsilon &= \varepsilon \\
f_s(x, w, r) &= xrx
\end{aligned}$$

Definiciones inductivas libres

Definición de $\mathbb{X}$

Definimos $\mathbb{X} \subseteq \mathbb{R}$ inductivamente con las siguientes reglas:

i $3 \in \mathbb{X}$

ii Si $x \in \mathbb{X}$, entonces $x - 2 \in \mathbb{X}$

iii Si $x \in \mathbb{X}, y \in \mathbb{X}$, entonces $x + y \in \mathbb{X}$

Preguntas

- ¿$3 \in \mathbb{X}$?
- ¿$1 \in \mathbb{X}$?
- ¿$6 \in \mathbb{X}$?

Definiciones Inductivas Libres

Definición inductiva libre

Una definición es libre cuando cada elemento del conjunto se forma de una única manera.

Definiciones no libres y esquemas de recursión

No deberíamos usar definiciones inductivas no libres para definir funciones.
Por ejemplo, si definimos $f : \mathbb{X} \to \mathbb{N}$ con las ecuaciones

$$\begin{aligned} f(3) &:= 0 \\ f(n-2) &:= 0 \\ f(x+y) &:= 1 + f(x) + f(y) \end{aligned}$$

¿ Cuánto vale $f(3)$?

Recursión General

Esquema de recursión primitiva

Para definir $f : A \to B$ se debe

- definir f para los objetos base de A, y
- definir f en los objetos obtenidos de aplicar cláusulas inductivas usando el valor de f en objetos *inmediatamente anteriores*

Esquema de recursión general

Para definir $f : A \to B$ se debe

- definir f para los objetos base de A, y
- definir f usando el valor de f obtenido para objetos *estrictamente menores*

Ejemplo de recursión general en $\mathbb{N}$

$$\begin{aligned} &\textsc{fibo} : \mathbb{N} \to \mathbb{N} \\ \textsc{fibo}(0) &= 1 \\ \textsc{fibo}(1) &= 1 \\ \textsc{fibo}(n+2) &= \textsc{fibo}(n) + \textsc{fibo}(n+1) \end{aligned}$$

Condiciones suficientes para definir una función

Exhaustividad Todo elemento del dominio debe computar a algún valor (totalidad)

No superposición Ningún elemento del dominio puede computar a más de un valor (propiedad funcional)

Terminación Las llamadas recursivas usan elementos menores (con respecto a un orden bien fundado) como argumentos

Ejemplo de recursión general en $\mathbb{N}$

$$
\begin{aligned}
\textsc{fibo} &: \mathbb{N} \to \mathbb{N} \\
\textsc{fibo}(0) &= 1 \\
\textsc{fibo}(1) &= 1 \\
\textsc{fibo}(n+2) &= \textsc{fibo}(n) + \textsc{fibo}(n+1)
\end{aligned}
$$

Condiciones suficientes

Exhaustividad Todo natural es cero, uno, o de la forma $n+2$; hay alguna regla que lo computa

No superposición Ser cero, uno, o de la forma $n+2$ son condiciones mutuamente incompatibles; es decir, cada cómputo está únicamente determinado

Terminación Usando el orden habitual tenemos que $n < n+2$ y $n+1 < n+2$

Ejemplo de recursión general en $\mathbb{N}$

$$\textsc{div} : \mathbb{N} \times \mathbb{N}^+ \to \mathbb{N}$$

$$\textsc{div}(n, m) = 0, \text{ si } n < m$$

$$\textsc{div}(n, m) = 1 + \textsc{div}(n - m, m), \text{ si } n \geq m$$

Condiciones suficientes

Exhaustividad Toda pareja $\langle n, m \rangle \in \mathbb{N} \times \mathbb{N}^+$ cumple $n < m$ o $m \leq n$

No superposición Ninguna pareja $\langle n, m \rangle \in \mathbb{N} \times \mathbb{N}^+$ cumple $n < m$ y $m \leq n$

Terminación Tenemos que $n - m < n$, y usamos el orden $\langle j, k \rangle \prec \langle j', k' \rangle := j < j'$

Resumen

Sea A un conjunto definido inductivamente.

- Si la definición *es libre*, se puede aplicar sin problemas el *esquema de recursión primitiva*.
- Si la definición *no es libre*, hay *superposición*. Hay que probar que los casos repetidos dan el mismo resultado.
- Si se usa un *esquema de recursión general* hay que probar *exhaustividad*, *no superposición* y *terminación*.